AMENDMENTS TO THE SPECIFICATION

Please amend the specification at the paragraphs indicated below such that the paragraphs of the specification at those indicated locations are as follows:

The paragraphs beginning at page 11, line 8 and carrying over to page 12, line 1:

Generally, the decoding process for a $\mathfrak{R}(2,4)$ code is known to workers skilled in the art, and is too involved to explain in detail herein. However, it is important to understand that, within the above-mentioned limits, any errors in the received encoded codeword can be overcome during the decoding process via majority voting. For example, ~~the codeword~~ a message coded in $\mathfrak{R}(2,4)$ can be expressed as a ~~sum of codewords~~ collection of codeword bits (c) as follows with the corresponding codeword formed from:

$$C = MG$$

where

$$M = [M_0 M_1 M_2]$$

and where

$$G = \begin{bmatrix} G_0 \\ G_1 \\ G_2 \end{bmatrix}.$$

Decoding of the message M begins in the Reed decoding algorithm by decoding M_2 and then proceeds to the decoding of M_1 and M_0 as is known by a worker skilled in the art.

For example, decoding of $M_2 = \{m_5, \ldots m_{10}\}$ leads to the following expressions:

$c_0 = m_0$

$c_1 = m_0 + m_4$

$c_2 = m_0 + m_3$

$c_3 = m_0 + m_3 + m_4 + m_{10}$

$c_4 = m_0 + m_2$

c~~5~~$_5 = m_0 + m_2 + m_4 + m_{10}$

c~~6~~$_6 = m_0 + m_2 + m_3 + m_8$

c~~7~~$_7 = m_0 + m_2 + m_3 + m_4 + m_8 + m_9 + m_{10}$

c~~8~~$_8 = m_0 + m_1$

c~~9~~$_9 = m_0 + m_1 + m_4 + m_7$

c~~10~~$_{10} = m_0 + m_1 + m_3 + m_6$

c~~11~~$_{11} = m_0 + m_1 + m_3 + m_4 + m_6 + m_7 + m_{10}$

c~~12~~$_{12} = m_0 + m_1 + m_2 + m_5$

c~~13~~$_{13} = m_0 + m_1 + m_2 + m_4 + m_5 + m_7 + m_9$

c~~14~~$_{14} = m_0 + m_1 + m_2 + m_3 + m_5 + m_6 + m_8$

~~$m_0 + m_1 + m_2 + m_3 + m_4 + m_5 + m_6 + m_7 + m_8$~~

~~$c15$~~ =

~~$m_9 + m_{10} + m_{11} + m_{12} + m_{13} + m_{14} + m_{15}$~~

$c_{15} = m_0 + m_1 + m_2 + m_3 + m_4 + m_5 + m_6 + m_7 + m_8$

$m_9 + m_{10} + m_{11} + m_{12} + m_{13} + m_{14} + m_{15}$

The paragraph beginning at page 13, line 17 and continuing through line 25:

As shown in FIG. 1, the ~~code rates~~ codeword lengths of Reed-Muller codes are plotted against the ~~codeword length~~ code rates. The four plotted lines represent four different minimum distances. When the graph line showing a minimum distance of 32 (d=32) is compared with the graph line showing a minimum distance of 4 (d=4), the trade-off between codeword length and code rate becomes apparent. Specifically, assuming a codeword length of 10,000 bits, the code rate of the code having a minimum distance of 32 is approximately 0.88, while the corresponding code rate for code having a minimum distance of 4 is approximately 0.99.

The paragraphs beginning at page 14, line 6 and carrying over to page 15, line 22:

FIG. 3 illustrates a communication ~~circuit~~ system 10 according to the present invention. The communication ~~circuit~~ system 10 includes an RM-encoder 12, a partial-response channel 14, and a Reed-Muller error correcting, partial response channel decoder 16 (RM-PR 16). For simplicity, a transmission of a single codeword *m* is shown, though the same principle of operation holds for every message word transmitted. As shown, codeword *m* is input into the RM encoder 12, which generates an RM-encoded codeword *x*. RM-encoded codeword *x* is transmitted over the Partial-Response (PR) channel 14, which tends to introduce noise to the codeword *x* such that the received codeword *y* is input into the RM-PR decoder 16, which produces an output decoded estimated codeword.

The RM encoder 12 accepts a message word *m* of *k*-binary digits (bits) and converts the message word *m* into a codeword *x* comprised of *n*-bits. The codeword *x* is then transmitted through a communication channel 14 (such as an optical fiber, a wireless communication channel, or any other type of communication channel) or recording channel 14 (such as magnetic hard disc drive, and the like). The channel 14 is equalized to a partial-response target. In other words, at decoder 16, the impulse response of the channel 14 to a ~~pre-defined~~ predefined transfer function is equalized to reduce the ~~effect of intersymbol interference~~ unwanted aspects of the channel response.

When the received codeword *y* arrives at the RM-PR decoder 16, the decoder 16 must attempt to optimally reconstruct the transmitted message word *m*. The decoder 16 ideally minimizes the number of bits where *m* and $\hat{m}$ differ so as to minimize the number of bit errors.

FIG. 4 shows a modified version of the communication ~~circuit~~ system 10 of FIG. 3 with an ~~exploded~~ internal subsystems view of the RM-PR decoder 16. Specifically, FIG. 4 illustrates a communication ~~circuit~~ system 10 including an RM encoder 12, a PR channel 14, a RM-PR decoder 16 and an interleaver 18 (denoted in FIG. 4 by the symbol $\pi^{[[-1]]}$) disposed between the RM encoder 12 and the PR channel 14. The interleaver 18 randomly ~~re-orders~~ reorders the RM-encoded codeword x prior to passing the codeword x to the partial-response channel 14. The interleaver 18 is ~~added to~~ shown here in the first portion of the communication circuit 10 in order to ~~highlight the position of~~ show a more complete system with respect to the corresponding elements in the RM-PR decoder 16[[,]] which is presented in an expanded form to show the decoder 16 in greater detail.

As shown, RM-PR decoder 16 has three stages A, B, and C. Stage A has a BCJR algorithm or SOVA algorithm based channel decoding device 20A, de-interleavers 22A (here shown as two ~~interleavers~~ de-interleavers denoted by π^{-1} to show that the corresponding two preceding outputs are being "de-interleaved"), and a Reed-Muller message passing device 24A. Stages B and C have corresponding BCJR or SOVA algorithm based channel decoding devices 20B and 20C, corresponding de-interleavers 22B and 22C, and corresponding Reed-Muller message passing devices 24B and 24C. Between stages A and B is an assembler 26A and an interleaver 28A, and between stages B and C is a corresponding assembler 26B and an interleaver 28B. Assemblers 26A and 26B and interleavers 28A and 28B perform a transitional function that converts the soft-decision data received from the corresponding preceding Reed-Muller message passing device into the proper form for the next stage of the decoder 16, and ~~are~~ such an assembler not necessary for the output of the final stage C of the decoder 16.

The paragraphs beginning at page 16, line 21 and carrying over to page 20, line 26::

Generally, the code bit decisions are represented in FIG. 4 as a vector $\hat{x}$ of length n, while the corresponding code bit reliabilities $\mu(\hat{x})$ are represented as vectors of the same length n. These two vectors are inputs to the Reed-Muller Message Passing Algorithm block 24. For the detected vector and the corresponding reliability vector, the RM-MPA block 24 produces a decoded message bit $\underline{\hat{m}}$ and the corresponding reliability vector $\mu(\hat{m})$.

Though the decoding process could ~~theoretically~~ stop at this point, but for nearer to optimum performance, one or more additional stages of processing ~~must occur~~ are chosen to be performed as described above. The decoded message bit $\hat{m}$ and the associated reliability vector $\mu(\hat{m})$ contain the decoded message bits and ~~information~~ the corresponding bit reliabilities, respectively, and not codeword reliabilities. The message bits and ~~information~~ the corresponding bit reliabilities must be converted ~~back~~ to ~~code words and code word~~ codeword reliabilities to be used with incoming information at the next processing stage. Assembler 26A ~~restores~~ converts the ~~information~~ message bit reliabilities $\mu(\hat{m})$ ~~into~~ to improved ~~code word~~ codeword reliability vectors which are then passed through an interleaver 28A ($\pi^{[[-1]]}$) to form $\lambda(\hat{x})$ that is provided to the next either BCJR or SOVA algorithm based channel decoding device 20B. The process is repeated n times (in this case, three times), and after n-stages, the last decoded message bits ~~becomes~~ become the output of RM-PR decoder 24C.

As shown, the either BCJR or SOVA algorithm based channel decoding devices 20A, 20B, and 20C are identical. The BCJR or SOVA algorithm based channel decoding device 20A has no initial reliability vector $\lambda(\hat{x})$ associated with the received from the channel information ~~code word~~ codeword y, as indicated above, and so the second input of the BCJR or SOVA device 20A is a reliability vector $\lambda(\hat{x})$ of value zero, providing no indication of reliability. Subsequent BCJR or SOVA devices 20B and 20C have improved reliability vectors $\lambda(\hat{x})$ at each subsequent stage, such that the final stage of processing has ~~an idealized~~ a significantly improved reliability vector $\lambda(\hat{x})$ with which to process the code bits of the received ~~code word~~ codeword y. Thus, the accuracy of the RM-PR decoder 16 improves with each iteration.

Generally, the RM-MPA block 24 uses an RM code (r,m) that has the following parameters: length $n=2^m$, dimension[[,]]

$$k = \sum_{i=0}^{r} \binom{m}{j},$$

rate $R=(n-k)/n$, and minimum distance 2^{m-r}, where the parameters $r>0$ and $m<l$ can be chosen arbitrarily. The notation

$$\binom{m}{j}$$

is the combinatorial combination or "m-choose-j" function (the number of ways of selecting j different objects from a set of m different objects). The RM generator matrix can be generally written as

$$G = \begin{bmatrix} G_0 \\ G_1 \\ \vdots \\ G_r \end{bmatrix}$$

wherein the ~~sub-matrix~~ submatrix G_j, $0£j£r$, is of size $\binom{m}{j} \times n$.

For the decoding of the RM codes, there exists an algorithm called the Reed algorithm as indicated above, which is a maximum likelihood decoding algorithm when only code bit hard decisions are available. In the present invention, the code bit reliabilities are utilized in making bit decisions instead of ~~the~~ just making code bit hard decisions.

Generally, in the Reed algorithm, the decoding is performed in r steps. In step j, the message word $x^{(j)}$ of length $\binom{m}{j}$, corresponding to G_j (with ~~indexes,)~~ message word bits $x_s^{(j)}$ having index s such that

$$\sum_{i=0}^{j-1}\binom{m}{i} + 1 \le s \le \sum_{i=0}^{j}\binom{m}{i})$$

is detected. For each of these bits, a set of 2^{m-j} linear equations can be written, involving all n bits of the received code word y. The positions of the code bits involved in the equations for calculating the s-th message bit, $x_s^{(j)}$, are given by the ~~array~~ corresponding one of arrays P_s that are based on the check sum equation and bit solution orthogonal to each bit $x_s^{(j)}$.

In each array P_s, each column shows the locations of the code bits to be added together to give an estimation of the message bit $x_s^{(j)}$. This addition is modulo-2 addition. In the hard-coding version, the message bit is determined by a majority vote among all 2^{m-j} estimations, (as described above), and the received code vector is updated according to the relation y [[=]]$\leftarrow$ y + $x^{(j)}G_j$, and then passed to the next decoding step (step j+1).

In the present invention, ~~instead of bit values,~~ the method uses the log-bit ~~likelihoods~~ liklihood ratios (LLR)[[.]] to indicate the bit reliabilities as

$$LLR(b_{[[k]]s}) = \log\left(\frac{Probability(b_{[[k]]s} = +1/y)}{Probability(b_{[[k]]s} = -1/y)}\right)$$

where $b_{[[k]]s}$ is the channel input and where y is the channel output. The log-bit likelihood of the *s-th* message bit from the *l-th* equation on the *j-th* step is calculated as follows, assuming all code bits are independent[[:]], as

~~where~~

$$\cancel{S_k^{i,j} = \prod_l sign\{\mu(x_l^{i,j,k})\}} \qquad \cancel{(1)}$$

$$\mu(\hat{m}_s^{i,j}) = -S_s^{i,j}.log\left\{-tanh\left(\frac{A_s^{i,j}}{2}\right)\right\}, \quad (1)$$

where $\mu\left(\hat{x}_{l}^{i,j,s}\right)$ is the LLR of the expression for the *s*-th message bit from the *i*-th equation in the *j*-th level, and where

~~and~~

$$\cancel{A_{k}^{i,j} = \sum_{l} \log\left|\tanh\left(\frac{\mu\left(\hat{x}_{l}^{i,j,k}\right)}{2}\right)\right|.} \qquad \cancel{(2)}$$

$$S_{s}^{i,j} = \prod_{l} sign\left\{\mu\left(\hat{x}_{l}^{i,j,s}\right)\right\} \qquad (2)$$

~~Where $\mu\left(\hat{x}_{l}^{i,j,k}\right)$ is the LLR of the expression for the *s*-th message bit from the *i*-th equation in the *j*-th level. The LLR can be expressed as follows:~~

and

$$\cancel{\mu\left(\hat{m}_{k}^{i,j}\right) = -S_{k}^{i,j}.log\left[-tanh\left(\frac{A_{k}^{i,j}}{2}\right)\right]} \quad \cancel{(3)}$$

$$A_{s}^{i,j} = \sum_{l} \log\left|\tanh\left(\frac{\mu\left(\hat{x}_{l}^{i,j,s}\right)}{2}\right)\right|. \qquad (3)$$

The product and summation operations in equations (2) and (3) above are performed on the bit likelihoods of the code bits whose locations in the received codeword are determined by the l-th column of the matrix P_s. For the reduced complexity, (1), (2), and (3) may be approximated as

~~$\mu(\hat{m}^{i,j}_{k}) = \left[\prod_{l} sign\{\mu(\hat{x}^{i,j,k}_{l})\}\right] \cdot \min_{l} \{|\mu(\hat{x}^{i,j,k}_{l})|\}. \qquad (4)$~~

$$\mu(\hat{m}^{i,j}_{s}) = \left[\prod_{l} sign\{\mu(\hat{x}^{i,j,s}_{l})\}\right] \cdot \min_{l} \{|\mu(\hat{x}^{i,j,s}_{l})|\}. \qquad (4)$$

The product and minimization operations in equation (4) are performed on the same bit likelihoods as in equations ([[1]]2) and ([[2]]3). Next, the updated LLR of the k-the message bit in the [[i]]j-th level is calculated by combining the likelihoods obtained from all 2^{m-j} equations. In other words

~~$\mu(\hat{m}_k) = \sum_{1 \le l \le 2^{m-j}} \mu(\hat{m}^{i,j}_{k}). \qquad (5)$~~

$$\mu(\hat{m}_s) = \sum_{1 \le l \le 2^{m-j}} \mu(\hat{m}^{i,j}_{s}). \qquad (5)$$

In the hard decoding version (Reed algorithm), a message bit is determined by a majority vote, and the received code vector is updated using the expression [[(2)]] $y\text{-}y+x^{(j)}G_j$ given above, and then passed to the next decoding step. In the soft decoding version, prior to advancing to the next row of the decoder 16, the updated message bit likelihoods must be converted to the code bit likelihoods. This conversion is performed based on the submatrix G_j.

FIG. 5 illustrates an eleven layer ~~illustration~~ representation of ~~the~~ a "Reed graph" designed to place the mathematical formulae of the instant invention into a graphical perspective though just shown for the code bits and messages bits formulae with a similar graph being used for the reliabilities formulae with corresponding indices. In this example, layer 1 represents a sixteen bit codeword received from the channel 14. In ~~reality~~ alternative situations, the codeword could be much larger than 16 bits.

In essence layer 1 represents the coded bits *y* received from the channel 16. ~~These~~ The equations given above for message bit m_{10}, are chosen as an example, and are ~~performed~~ determined at layer 2 in FIG. 5 for reduced complexity.

The paragraphs beginning at page 21, line 4 and carrying over to page 22, line 16:

The positions of ones in the *i-th* column of G_j (below) indicate what message bits on the *j-th* level are used to calculate the *i-th* code bit in accordance with the codeword generation matrix equation. For example, the nonzero elements in the 15-th column of G_2 below are 1, 2, and 3. This means that on the second level, the message bits [[m6]]m_6 (=1+5), [[m7]]m_7 (=2+5) and [[m8]]m_8 (=3+5) are used to calculate the 15-th code bit. On the second level of encoding in this example, only the message bits from the location 6 to 11 are used again in accord with the codeword generator matrix equation. In a similar fashion we can find which message bits are used to compute *i-th* bit on the *j-th* level.

Suppose that the nonzero positions of the *l-th* column of G_j are given by the set $\{l_1 \ldots l_t\}$ where

$$l_1 \geq \sum_{i=0}^{j-1} \binom{m}{i} \quad \text{and} \quad l_t \leq \sum_{i=0}^{j} \binom{m}{i}$$

Then the conversion ~~form~~ from message bit LLR to ~~codebit~~ code bit LLR ($\mu \rightarrow \lambda$) ~~can be done as follows:~~

is provided as

$$\lambda(\hat{x}_l) = -S_l \cdot \log\left\{-\tanh\left(\frac{A_l}{2}\right)\right\} \qquad (6)$$

where

$$S_l = \coprod_j sign\{\mu(\hat{m}_{l_j})\} \qquad [[(6)]](7)$$

and

$$A_l = \sum_j log\left|tanh\left(\frac{\mu(\hat{m}_{l_j})}{2}\right)\right|. \qquad [[(7))(8)$$

The equations (6), (7), and (8) may also be approximated as follows.

$$\cancel{\lambda(\hat{x}_l) = \left[\prod_j sign\{\mu(\hat{m}_{l_j})\}\right] \cdot \underset{j}{Min} \left|\{\mu(m_{l_j})\}\right| \qquad (8)}$$

$$\lambda(\hat{x}_l) = \left[\prod_j sign\{\mu(\hat{m}_{l_j})\}\right] \cdot \min_j \{|\mu(m_{l_j})|\} \qquad (9)$$

This processing is performed layer 4 in the Reed graph of FIG. 5.

The updated LLR's for all of the ~~codebits~~ code bits, which are to be used in the next decoding step, are found by applying equations (1), (2), and (3), or (4) above to the output of ~~equation (8) and~~ equations (6) or (9) as the output of the previous decoding level (*j*-1), which is the same as the input to the current decoding level (j). The final ~~function of the~~ processing for each of the first two stages of RM-PR decoder 16 is provided by the ~~assembler~~ corresponding one of assemblers 26A[[,]] and 26B for each stage respectively. This assembler function is performed by again applying equations (6), (7) and (8) or (9) to the LLR sequence of reliabilities of decoded message bits $\mu(\hat{m})$, and this time the entire generator matrix G is used. The above equations (1-[[8]]9) and subsequent updates ~~codebit~~ to the code bit log likelihood reliabilities can be viewed as providing a message passing algorithm on ~~a special type of graph called a~~ the [["]]Reed graph[["]] indicated above.

Then the process is repeated using the soft-decoded reliability vector derived from ~~equation (11)~~ equations 6 or 9 above beginning with layer 6 in the Reed graph of Figure 5, which uses the original 16 bits and the reliability vector.

The paragraphs beginning at page 22, line 22 and carrying over to page 25, line 6:

To illustrate the idea, we use the following example. Consider the (r,m)=(2,4) RM code. The corresponding generator sub-matrices are ~~given~~ repeated below [[.]] from above as

$$G_0 = \begin{bmatrix} 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \end{bmatrix},$$

$$G_1 = \begin{bmatrix} 0&0&0&0&0&0&0&0&1&1&1&1&1&1&1&1 \\ 0&0&0&0&1&1&1&1&0&0&0&0&1&1&1&1 \\ 0&0&1&1&0&0&1&1&0&0&1&1&0&0&1&1 \\ 0&1&0&1&0&1&0&1&0&1&0&1&0&1&0&1 \end{bmatrix},$$

$$G_2 = \begin{bmatrix} 0&0&0&0&0&0&0&0&0&0&0&0&1&1&1&1 \\ 0&0&0&0&0&0&0&0&0&0&1&1&0&0&1&1 \\ 0&0&0&0&0&0&1&1&0&0&0&0&0&0&1&1 \\ 0&0&0&0&0&0&0&0&0&1&0&1&0&1&0&1 \\ 0&0&0&0&0&1&0&1&0&0&0&0&0&1&0&1 \\ 0&0&0&1&0&0&0&1&0&0&0&1&0&0&0&1 \end{bmatrix}.$$

~~From the~~ The four equations ~~(from above), and the arrays of orthogonal parity check positions~~ for the message bit m_{10} given above are repeated here:

$$
\begin{aligned}
m_{10} &= c_0 + c_1 + c_2 + c_3 && (10)\\
m_{10} &= c_4 + c_5 + c_6 + c_7 && (11)\\
m_{10} &= c_8 + c_9 + c_{10} + c_{11} && (12)\\
m_{10} &= c_{12} + c_{13} + c_{14} + c_{15} && (13)
\end{aligned}
$$

~~It is possible to build an~~ A corresponding array [[or]] of the orthogonal parity checks for the message bit m_{10} [[as]] is shown below[[.]] ($P_5 = P_{10}$): ~~Note that the first column of P_{10} is a set of indices from the first expression (10) above, and the other columns represent sets of indices from expressions 11-13.~~

$$
P_{10} = \begin{bmatrix} 0 & 4 & 8 & 12 \\ 1 & 5 & 9 & 13 \\ 2 & 6 & 10 & 14 \\ 3 & 7 & 11 & 15 \end{bmatrix}
$$

Note that the first column of P_{10} is a set of indices from the first expression (10) above, and the other columns represent sets of indices from expressions (11)-(13).

~~$P_{10} = \begin{bmatrix} 0 & 4 & 8 & 12 \\ 1 & 5 & 9 & 13 \\ 2 & 6 & 10 & 14 \\ 3 & 7 & 11 & 15 \end{bmatrix}$~~

Similarly, as indicated above, for the other message bits $(\hat{m}_l)$ the corresponding ones of arrays P_s involving the related checksum equations and bit solutions orthogonal thereto are

$$P_9 = \begin{bmatrix} 0 & 2 & 8 & 10 \\ 1 & 3 & 9 & 11 \\ 4 & 6 & 12 & 14 \\ 5 & 7 & 13 & 15 \end{bmatrix} \qquad P_8 = \begin{bmatrix} 0 & 1 & 8 & 9 \\ 2 & 3 & 10 & 11 \\ 4 & 5 & 12 & 13 \\ 6 & 7 & 14 & 15 \end{bmatrix}$$

$$P_5 = \begin{bmatrix} 0 & 4 & 8 & 12 \\ 1 & 5 & 9 & 12 \\ 2 & 6 & 10 & 14 \\ 3 & 7 & 11 & 15 \end{bmatrix} \quad P_6 = \begin{bmatrix} 0 & 1 & 4 & 5 \\ 2 & 3 & 6 & 7 \\ 8 & 9 & 12 & 13 \\ 10 & 11 & 14 & 15 \end{bmatrix} \quad P_7 = \begin{bmatrix} 0 & 2 & 4 & 6 \\ 1 & 3 & 5 & 7 \\ 8 & 10 & 12 & 14 \\ 9 & 11 & 13 & 15 \end{bmatrix}$$

$$P_4 = \begin{bmatrix} 0 & 2 & 4 & 6 & 8 & 10 & 12 & 14 \\ 1 & 3 & 5 & 7 & 9 & 11 & 13 & 15 \end{bmatrix}$$

$$P_3 = \begin{bmatrix} 0 & 1 & 4 & 5 & 8 & 9 & 12 & 13 \\ 2 & 3 & 6 & 7 & 10 & 11 & 14 & 15 \end{bmatrix}$$

$$P_2 = \begin{bmatrix} 0 & 1 & 2 & 3 & 8 & 9 & 10 & 11 \\ 4 & 5 & 6 & 7 & 2 & 3 & 4 & 5 \end{bmatrix}$$

$$P_1 = \begin{bmatrix} 0 & 1 & 2 & 3 & 4 & 5 & 6 & 7 \\ 8 & 9 & 10 & 11 & 12 & 13 & 14 & 15 \end{bmatrix}$$

$$P_0 = [0 \;\; 1 \;\; 2 \;\; 3 \;\; 4 \;\; 5 \;\; 6 \;\; 7 \;\; 8 \;\; 9 \;\; 10 \;\; 11 \;\; 12 \;\; 13 \;\; 14 \;\; 15]$$

The Reed graph of Figure 5, as indicated above, is based on the above parity check arrays ~~is illustrated in FIG. 5~~ and illustrates the determination of selected ones of the message bits. The bipartite graph in the top portion of the figure corresponds to the orthogonal parity checks (squares) involving the second level message bits and code bits (j=2). The connections involving the four parity checks in the top portion are defined by the corresponding checksum equation and bit solution for message bit m_{10} as reflected in the array [[P10]] P_{10}. All other connections for the second level message bits ~~m5,m6,...,m9~~ $m_5, m_6,...,m_9$ can be found in the corresponding arrays ~~Pf,P6,...,P9~~ $P_5,P_6,...,P_9$.

In the first step of decoding, the code bit LLR's (layer 1) are passed through the check nodes (layer 2) to the message variables (layer 3), using equations (1), (2), and (3) or equation (4), and the equation (5), from above. After computing all the message bit likelihoods on the second level, submatrix G2 is used to update the code bit likelihoods using equations ~~(5), (6), and (7)~~ (6), (7), and (8) or equation [[(8)]] (9) above. The first iteration step is finished when all message bits and all message bit likelihoods are obtained and all code bit likelihoods are updated. Then, decoding proceeds to the next iteration.